Project Documentation

Jelly Space

By

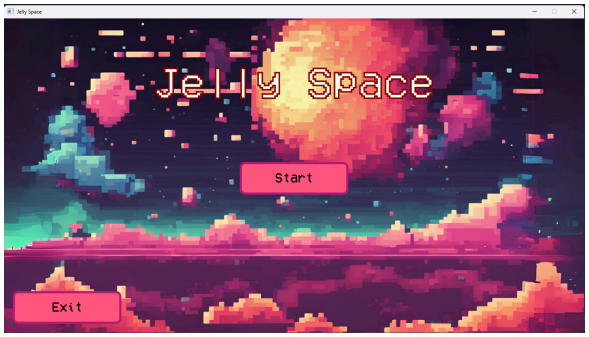
Chatrin Yoonchalard
6631304721
Chayaphon Kultanon
6631306021
(Group : Jelly)

2110215 Programming Methodology
Semester 2 Year 2024
Chulalongkorn University

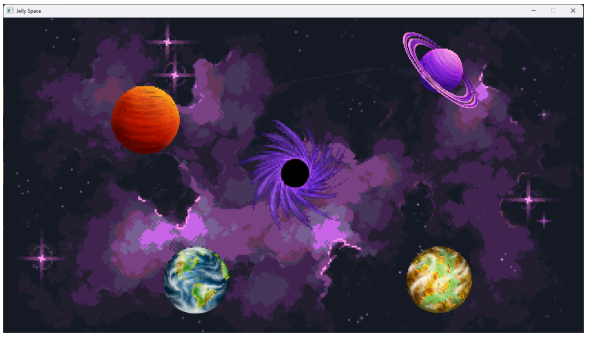
What is Jelly Space? And how to play it?

What is Jelly Space?

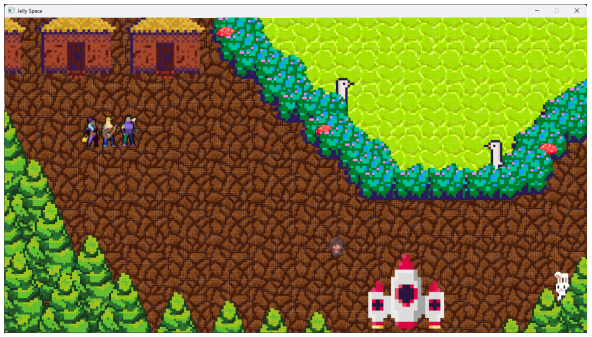
Jelly Space is an exciting and adventurous turn-based strategy game set in a vibrant galaxy of unique planets. Each planet, or map, features a distinct theme and challenges players to conquer it by defeating powerful bosses. Your mission is to travel across these maps, battle formidable foes, and ultimately restore peace to the cosmos by vanquishing all the bosses.



(StartPane)



(MapSelectPane)



(MapPane)

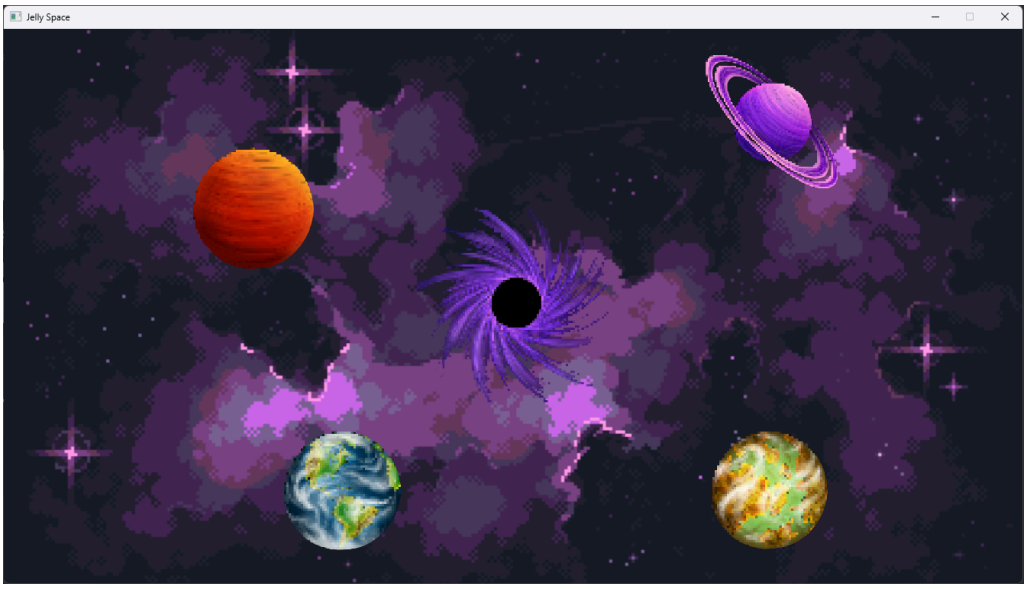
(BattlePane)

How to Play Jelly Space

Objective: Conquer all five space-themed maps by defeating the boss on each map.

Gameplay Overview:

1. Exploration and Navigation:
   - Start by selecting a map to explore. Each map has a distinct theme and a unique boss.
   - Navigate through the map, encountering various challenges and preparing for the final showdown with the boss.



(MapSelectPane)

2. Battle System:
   - Engage in turn-based battles similar to Pokémon. Each turn, you and the boss will take actions such as attacking, guarding, or using abilities.
   - Players control a team of monsters, each with its own set of attacks, guards, and unique abilities. Some monsters may lack certain actions, adding a layer of strategy to team composition.

(BattlePane)

3. Using Potions:

- During battles, players can use potions to heal their monsters or provide other benefits. Each potion can be used once per map. However, if you are defeated by the boss, the potions will be available for use again on your next attempt. Strategically managing your potion inventory is crucial for overcoming tough opponents.

4. Victory and Defeat:

- To win a battle, reduce the boss's HP to zero. Upon victory, the map is conquered.
- If all your monsters are defeated, you lose the battle. You can either attempt the battle again or explore a different map to strengthen your strategy.

Tips for Success:

- Strategize Your Team: Carefully choose which monsters to bring into battle based on their abilities and the boss's attack pattern .
- Manage Resources: Use potions wisely and consider when to attack, guard, or use special abilities.
- Learn and Adapt: Each boss has unique strategies. Learn their patterns and adapt your tactics accordingly.

Winning the Game:

- Successfully conquer all five maps by defeating the bosses on each one to complete the game and achieve victory in Jelly Space.

(CongratulationPane)

Notes on UML & Jar

Because the UML diagram is too large, the UML is saved as .svg file It is recommended that the jar file (Jelly_Space.jar) be run with JavaFX 22 (or at least JavaFX 21) as the application is developed with JavaFX 22. Here is drive for Jar file and UML

https://drive.google.com/drive/folders/1uvOnpar4JA6-S0kbT1Om6w0IysFNWkql?usp=sharing

Implementation Details

Noted that Access Modifier Notations can be listed below:

+ (public), # (protected), - (private), underlined (static), ALL_CAPS (final variable),

***italic* (abstract)**

1. Package boundaries

1.1 Abstract Class Boundary

Methods

Name	**Description**
+ boolean isWithinBoundary(double x,double y)	Checks if the given coordinates (x, y) are within the boundary.
+ void drawBoundary(GraphicsContext gc)	Draws the boundary on the provided GraphicsContext.

1.2 Class EllipseBoundary extends Boundary

Fields

Name	**Description**
- double centerX	The x-coordinate of the center of the

	ellipse.
- double centerY	The y-coordinate of the center of the ellipse.
- double radiusX	The horizontal radius of the ellipse.
- double radiusY	The vertical radius of the ellipse.

Constructor

Name	**Description**
+ EllipseBoundary(double centerX, double centerY, double radiusX, double radiusY) + boolean isWithinBoundary(double x, double y)	Constructor that initializes the center coordinates and radii of the ellipse.

Methods

Name	**Description**
+ boolean isWithinBoundary(double x, double y)	Checks if the given coordinates (x, y) are within the boundary of the ellipse.
+ void drawBoundary(GraphicsContext gc)	Draws the ellipse boundary on the provided GraphicsContext.

1.3 Class InverseParabolicBoundary extends Boundary

Fields

Name	**Description**
- double h	The x-coordinate of the vertex of the inverse parabolic boundary.
- double k	The y-coordinate of the vertex of the inverse parabolic boundary.
- double c	The constant that determines the curvature of the inverse parabolic boundary.

Constructor

Name	**Description**
+ InverseParabolicBoundary(double h, double k, double c)	**Constructor that initializes the vertex coordinates (h, k) and the curvature constant (c) of the inverse parabolic boundary.**

Methods

Name	Description
+ boolean isWithinBoundary(double x, double y)	Checks if the given coordinates (x, y) are within the inverse parabolic boundary defined by the equation (y - k) >= -c*(x - h)^2
+ void drawBoundary(GraphicsContext gc)	Draws the inverse parabolic boundary on the provided GraphicsContext by plotting points along the curve and connecting them with lines. The boundary is drawn in red with a line width of 2.

1.4 Class LinearBoundary extends Boundary

Fields

Name	Description
- double m	The slope of the linear boundary.
- double c	The y-intercept of the linear boundary.

Constructor

Name	Description

+ LinearBoundary(double m, double c)	Constructor that initializes the slope (m) and y-intercept (c) of the linear boundary.

Methods

Name	**Description**
+ boolean isWithinBoundary(double x, double y)	Checks if the given coordinates (x, y) are within the linear boundary defined by the equation $y \leq m \cdot x + c$
+ void drawBoundary(GraphicsContext gc)	Draws the linear boundary on the provided GraphicsContext by plotting a line from (0 , c) to (1280 , m · 1280+c). The boundary is drawn in cyan.

1.5 Class ParabolicBoundary extends Boundary

Fields

Name	Description
- double h	The x-coordinate of the vertex of the parabolic boundary.
- double k	The y-coordinate of the vertex of the parabolic boundary.
- double c	The coefficient that defines the "width" and direction of the parabola.

Constructor

Name	Description
+ ParabolicBoundary(double h, double k, double c)	Constructor that initializes the vertex coordinates (h, k) and the coefficient (c) of the parabolic boundary.

Methods

Name	Description

+ boolean isWithinBoundary(double x, double y)	Checks if the given coordinates (x, y) are within the parabolic boundary defined by the equation (y - k) <= - c * (x - h)^2
+ void drawBoundary(GraphicsContext gc)	Draws the parabolic boundary on the provided GraphicsContext. It plots points along the parabola and connects them with lines, starting from x=0 to x=1280. The boundary is drawn in red with a line width of 2.

1.6 Class RectangleBoundary extends Boundary

Fields

- double x1	The x-coordinate of the first corner of the rectangle.
- double y1	The y-coordinate of the first corner of the rectangle.
- double x2	The x-coordinate of the second corner of the rectangle.

- double y2	The y-coordinate of the second corner of the rectangle.

Constructor

+ RectangleBoundary(double x1, double y1, double x2, double y2)	Constructor that initializes the coordinates of the two corners of the rectangle.

Methods

+ boolean isWithinBoundary(double x, double y)	Checks if the given coordinates (x, y) are within the rectangle defined by the two corners (x1, y1) and (x2, y2).
+ void drawBoundary(GraphicsContext gc)	Draws the rectangle boundary on the provided GraphicsContext. It uses the coordinates of the two corners to draw a rectangle outline. The boundary is drawn in cyan color.

2. Package entities

2.1 Package Monster

2.1.1 Package Abilities

2.1.1.1 Interface Attackable

Methods

Name	**Description**
boolean attack(Base_Monster otherMonster)	Defines the behavior of an attack action on another Base_Monster. Returns true if the attack is successful, false otherwise.
String getAttack()	Retrieves the type of attack.

2.1.1.2 Interface Guardable

Methods

Name	**Description**
boolean guard(Base_Monster ChosenMonster)	boolean guard(Base_Monster ChosenMonster)

2.1.1.3 Interface Unique_Ability

Methods

Name	Description
boolean unique_ability(Base_Monster monster)	Specifies the unique ability of a monster when invoked. Returns true if the unique ability is successfully activated, false otherwise.
String getUnique()	Retrieves a string representation of the unique ability.

2.1.2 Abstract class Base_Monster extends Sprite

Fields

Name	Description
- String name	Name of the monster.
- int maxHp	Maximum hit points of the monster.
- int hp	Current hit points of the monster.
- int maxMana	Maximum mana points of the monster.
- int mana	Current mana points of the monster.
- int baseDmg	Base damage of the monster.

- int dmg	Current damage of the monster.
- int baseDef	Base defense of the monster.
- int def	Current defense of the monster.
- int manaReg	Mana regeneration rate of the monster.
- boolean owned	Indicates whether the monster is owned by the player.
- Image dead_img	Image representing the dead state of the monster.
- Image idle_ally_img	Image representing the idle state of the monster in an ally view.
- Image idle_battle_img	Image representing the idle state of the monster in a battle view.
- Image special_ally_img	Image representing a special state of the monster in an ally view.

Constructor

Name	**Description**
+ Base_Monster(String name,int maxHp,int maxMana,int manaReg,int baseDmg,int baseDef,boolean owned, double x, double y, double width, double height, double speed, Image img)	Constructor for Base Monster class.

Methods

Name	**Description**
+ boolean isDead()	Checks if the monster is dead.
+ String toString(String amount,String type,int point,String x)	Converts the specified parameters into a formatted string.
+ String Choose_Boss_Ability(Base_Monster monster)	Converts the specified parameters into a formatted string.
+ void startTurn()	Gets the name of the monster.
+ getters and setters	Gets the maximum HP of the monster.

2.1.3 Class Chatrin extends Base_Monster implements Attackable, Guardable, Unique_Ability

Fields

Name	Description
- Image img	idle image of Chatrin

Constructor

Name	Description
+ Chatrin(int x,int y)	Constructor for Chatrin

Methods

Name	Description
+ boolean attack(Base_Monster otherMonster)	attack boss
+ String getAttack()	get a string of how attack work
+ boolean guard(Base_Monster ChosenMonster)	raise Chatrin Def
+ boolean unique_ability(Base_Monster monster)	Chatrin’s Unique Ability

+ String getUnique()	get a string of how unique ability work

2.1.4 Class Fai extends Base_Monster implements Attackable, Unique_Ability

Fields

Name	**Description**
- Image img	Idle image of Fai

Constructor

Name	**Description**
+ Fai(int x, int y, boolean owned)	Constructor of Fai

Methods

Name	**Description**
+ boolean attack(Base_Monster otherMonster)	attack boss and decrease boss mana
+ String getAttack()	get a string of how attack work

+ boolean unique_ability(Base_Monster monster)	Fai’s Unique ability
+ String getUnique()	get a string of how unique ability work

2.1.5 Class Faith extends Base_Monster implements Attackable, Guardable, Unique_Ability

Fields

Name	**Description**
- Image img	Idle image of Faith

Constructor

Name	**Description**
+ Faith(int x, int y)	Constructor of Faith

Methods

Name	**Description**
+ boolean attack(Base_Monster otherMonster)	attack the player’s monster
+ String getAttack()	null

+ boolean guard(Base_Monster ChosenMonster)	increase boos' DEF
+ boolean unique_ability(Base_Monster monster)	AOE damage to player's monsters
+ String getUnique()	null

2.1.6 Class Fei extends Base_Monster implements Unique_Ability

Fields

Name	**Description**
- Image img	Idle image of Fei

Constructor

Name	**Description**
+ Fei(int x, int y, boolean owned)	Constructor of Fei

Methods

Name	**Description**
+ boolean unique_ability(Base_Monster monster)	Sacrifice HP for damage
+ String getUnique()	get a string of how unique ability work

2.1.6 Class TU_Force extends Base_Monster implements Attackable, Unique_Ability

Fields

Name	**Description**
- Image img	Idle image of TU_Force

Constructor

Name	**Description**
+ TU_Force(int x, int y, boolean owned)	Constructor of TU_Force

Methods

Name	**Description**
+ boolean attack(Base_Monster otherMonster)	attack active monster
+ String getAttack()	null
+ boolean unique_ability(Base_Monster monster)	AOE damage
+ String getUnique()	null

2.2 Package Player

2.2.1 Class Inventory

Fields

Name	**Description**
- ArrayList<Base_Item> Items	ArrayList to collect player's items
- int MAX_ITEMS	Maximum item's that player can hold

Constructor

Name	**Description**
+ Inventory()	Constructor of Inventory

Methods

Name	**Description**
+ void addItem(Base_Item item)	add the item to inventory
+ ArrayList<Base_Item> getItems()	get ArrayList

2.2.2 Class My_Monster

Fields

Name	**Description**
- ArrayList<Base_Monster> monsters	ArrayLIst to collect monsters of players
- int MAX_MONSTER	Maximum monsters that player can hold

Constructor

Name	**Description**
+ My_Monster()	Constructor for My_Monster

Methods

Name	**Description**
+ void addMonster(Base_Monster monster)	add monster to ArrayList
+ ArrayList<Base_Monster> getMonsters()	get ArrayList of monster

2.2.3 Class Player extends Sprite

Fields

Name	**Description**
- String name	Name of the player.

- My_Monster my_monster	The player's monster.
- Inventory inventory	The player's inventory.
- int ACTION_POINT	Constant representing the player's action points.
- int Used_Point	The number of action points used by the player.
- Player player	Singleton instance of the player.
- int WIDTH	Width of the player.
- int HEIGHT	Height of the player.
- Image IMG_RIGHT	Image of the player facing right.
- Image IMG_LEFT	Image of the player facing left.
- double newX,newY	New position coordinates for the player.
- Rectangle playerRect, bossRect, rocketRect	Rectangles representing the player, boss, and rocket for collision detection.
- Base_Monster activeMonster	The currently active monster controlled by the player.

Constructor

Name	**Description**
+ Player(String name,double x, double y, double width, double height)	Constructor for the Player class with specified name, position, width, and height.

Methods

Name	**Description**
+ void update()	Updates the player's movement and checks for collisions with boss and rocket.
+ void movePlayer()	Moves the player based on keyboard input and checks for collisions.
+ void draw(GraphicsContext gc)	Draws the player on the canvas.
+ void drawBoundary(GraphicsContext gc)	Draws boundaries for the player, boss, and rocket for debugging purposes.
+ void createEntitiesBound()	Creates rectangles for the player, boss,

	and rocket.
+ getters and setters	getters and setters of all field

2.2.4 Class Rocket extends Sprite

Fields

Name	**Description**
- Image rocket	Image of the rocket.
- Image purple_rocket	Image of the purple rocket.
- Image red_rocket	Image of the red rocket.

Constructor

Name	**Description**
+ Rocket(double x,double y,String rocketColor)	Constructor for the rocket with the specified position and rocket color.

Methods

Name	**Description**
+ void setRocket(String color)	Sets the rocket image based on the

	specified color.

2.3 Class Sprite

Fields

Name	**Description**
- double x	X position of the sprite.
- double y	Y position of the sprite.
- double width	Width of the sprite.
- double height	Height of the sprite.
- double speed	Speed of the sprite.
- Image image	Image of the sprite.

Constructor

Name	**Description**

+ Sprite(double x, double y, double width, double height,double speed,Image img)	Constructor for the sprite with the specified position, size, speed, and image.

Methods

Name	**Description**
+ void move(double newX, double newY)	Moves the sprite to the specified position.
+ getters and setters	getters and setters of all field

3. Package gui

3.1 Package battle

3.1.1 Class ActionPane extends GridPane

Field

Name	**Description**
# ActionPane instance	A protected static instance of the ActionPane class to implement the singleton pattern.
# Text actionText	A protected Text element to display the current action being performed (e.g., Attack,

	Guard).
# Text itemDetail	A protected Text element to display details about the selected item.
- Text actionPoint	A private Text element to display the current action points of the player.
# ImageView attackButton	A protected ImageView for the attack button, which triggers the attack action.
# ImageView GuardButton	A protected ImageView for the guard button, which triggers the guard action.
# ImageView UniqueButton	A protected ImageView for the unique ability button, which triggers the unique ability action.
# MonsterDetail monsterDetail	A protected MonsterDetail component displaying details of the currently selected monster.
- Base_Monster boss	A private Base_Monster representing the boss that the player is fighting against on the current map.
- int cellHeight	A private integer representing the height of each cell in the grid layout. It is set to 78.
- int cellWidth	A private integer representing the width of

	each cell in the grid layout. It is set to 220.

Constructor

Name	**Description**
+ ActionPane()	Constructor. Initializes the ActionPane instance, sets up the UI components, and assigns the boss from the current game map.

Methods

Name	**Description**
+ void init()	Initializes the layout and design of the ActionPane, including setting up column and row constraints, background, border, buttons, texts, and spacing.
+ void createText()	Sets up the action and item detail texts, configures their fonts and colors, adds them to the layout, and creates a MonsterDetail component.
+ void createButton()	Creates and configures the attack, guard, and unique ability buttons, including setting their

	images and mouse event handlers for clicking and hovering.
+ void createActionPoint()	Creates and updates the action point display text. Removes the existing action point text if present and adds the updated text to the layout.
+ void handleHover(Base_Monster monster, String method)	Updates the action text based on the given monster's capabilities (attack, guard, or unique ability) when a button is hovered over.
+ void handleAttack()	Handles the attack action when the attack button is clicked. Checks if the active monster can attack, performs the attack on the boss, and updates the battlefield with the action text.
+ void handleGuard()	Handles the guard action when the guard button is clicked. Checks if the active monster can guard, performs the guard action, and updates the battlefield with the action text.

+ void handleUnique()	Handles the unique ability action when the unique button is clicked. Performs the unique ability, updates the battlefield with the action text, and temporarily changes the active monster's image.
+ void update()	Updates the action point display by calling the createActionPoint method.
+ void setItemDetail(String detail)	Sets the item detail text to the provided detail string.
+ void setMonsterDetail(MonsterDetail monsterDetail)	Replaces the current MonsterDetail component with the provided one and updates the layout.
+ ActionPane getInstance()	Returns the singleton instance of ActionPane.

3.1.2 Class BattleFieldPane extends Pane

Field

Name	**Description**
# BattleFieldPane instance	A protected static instance of the BattleFieldPane class to implement the

	singleton pattern.
# Base_Monster myMonster	A protected Base_Monster representing the player's active monster.
# Base_Monster enemyMonster	A protected Base_Monster representing the enemy (boss) monster.
# MonsterDetail myMonsterDetail	A protected MonsterDetail component displaying details of the player's active monster.
# MonsterDetail enemyMonsterDetail	A protected MonsterDetail component displaying details of the enemy monster.
- Canvas battleCanvas	A private Canvas for drawing the battle scene.
- GraphicsContext gc	A private GraphicsContext for drawing on the battleCanvas.
- Text battleLog	A private Text element for displaying battle logs.
- Image activeMonsterImage	A private Image representing the player's active monster.
- Image enemyMonsterImage	A private Image representing the enemy monster.
- double activeMonsterPosX	A private double representing the X position

	of the player's active monster on the canvas. It is set to 100.
- double activeMonsterPosY	A private double representing the Y position of the player's active monster on the canvas. It is set to 50.
- double bossPosX	A private double representing the X position of the enemy monster on the canvas. It is set to 850.
- double bossPosY	A private double representing the Y position of the enemy monster on the canvas. It is set to 50.
- long lastLogTime	A private long representing the timestamp of the last battle log.

Constructor

Name	**Description**
+ BattleFieldPane()	Constructor. Initializes the BattleFieldPane instance, sets up the UI components, and draws the initial battle scene.

Methods

Name	Description
+ void init()	Initializes the layout and design of the BattleFieldPane, including setting background, borders, player and enemy monster images, and creating monster details and battle canvas.
+ void createMonsterDetail()	Creates and configures the MonsterDetail components for both the player's and enemy's monsters, and adds them to the pane.
+ void update()	Updates the BattleFieldPane by checking the player's monster status, updating monster details, and refreshing the display.
+ void draw()	Draws the player's and enemy's monsters on the battleCanvas.
+ void handleBattle(String detail)	Handles the battle log display by creating a new log text, adding it to the pane, and setting a timer to remove it after a specified duration.
Getter & Setter	Getters and Setters for all fields

3.1.3 Class BattlePane extends GridPane

Field

Name	**Description**
# BattlePane instance	A protected static instance of the BattlePane class to implement the singleton pattern.
# Thread battleLoop	A protected Thread that runs the main battle loop.
# volatile boolean gameEnd	A protected volatile boolean indicating if the game has ended.
- AtomicBoolean isBossTurnStarted	A private AtomicBoolean indicating if the boss's turn has started.
# Boolean turn	A protected Boolean indicating the current turn. true for player's turn, false for boss's turn.
# ActionPane actionPane	A protected ActionPane for player actions.
# BattleFieldPane battleFieldPane	A protected BattleFieldPane displaying the battlefield.
# InventoryPane inventoryPane	A protected InventoryPane displaying the player's inventory.
# MonsterPane monsterPane	A protected MonsterPane displaying the player's monsters.

- int cellWidth	A private integer representing the width of each grid cell, set to 250.
- int cellHeight	A private integer representing the height of each grid cell, set to 128.
- MediaPlayer mediaPlayer	A private MediaPlayer for playing background music during battles.

Constructor

Name	**Description**
+ BattlePane()	Constructor. Initializes the BattlePane instance, sets up the UI components, and starts the battle loop.

Methods

Name	**Description**
+ void init()	Initializes the layout and design of the BattlePane, including setting background, grid constraints, creating child panes, and playing background music.
+ void startBattle()	Starts the battle loop in a separate thread,

	updating and drawing the battlefield at approximately 60 FPS, and checks the game state.
+ void endBattle(Boolean win)	Handles the end of the battle by resetting game state, showing win/lose messages, and transitioning back to the map or congratulations scene.
+ void draw()	Draws the battlefield and updates the action pane.
+ void update()	Updates the battlefield and manages turn-based actions.
+ void checkGameState()	Continuously checks the game state in a separate thread to determine if the player or boss is defeated or if the player's turn has ended.
+ void startBossTurn()	Initiates the boss's turn in a separate thread, executing boss abilities and transitioning back to the player's turn.
+ void setPlayerTurn(Boolean turn)	Sets the turn to either the player or the boss and performs necessary updates for the start of the turn.
+ void playMusic()	Plays background music for the battle and

	manages the volume transition for smooth audio playback.
Getters & Setters	Getters and Setters for all fields

3.1.4 Class InventoryPane extends GridPane

Field

Name	**Description**
# InventoryPane instance	A protected static instance of the InventoryPane class to implement the singleton pattern.

Constructor

Name	**Description**
+ InventoryPane()	Constructor. Initializes the InventoryPane instance and sets up the UI components.

Methods

Name	**Description**
+ void init()	Initializes the layout and design of the

	InventoryPane, including setting background, grid constraints, loading inventory items, and adding them to the pane.
+ void handleItem(Base_Item item)	Handles the logic for using an item from the inventory. If the item is a potion, it is used on the active monster. If it is a poison, it is used on the enemy boss.
+ void handleHover(Base_Item item)	Handles the logic for displaying item details in the ActionPane when the mouse hovers over an item.
+ static InventoryPane getInstance()	Returns the singleton instance of InventoryPane.
+ void update()	Updates the inventory pane by changing the images of used items and setting their mouse event handlers.
+ void resetItem()	Resets all items in the inventory to their unused state.

3.1.5 Class MonsterDetail extends VBox

Field

Name	Description
# Text monsterName	A protected Text field representing the monster's name.
# Text monsterDamage	A protected Text field representing the monster's damage value.
# Text monsterDef	A protected Text field representing the monster's defense value.
# Text monsterHp	A protected Text field representing the monster's health points.
# Text monsterMana	A protected Text field representing the monster's mana points.

Constructor

Name	Description
+ MonsterDetail(String monsterName, String monsterDamage, String monsterDef, String monsterHp, String monsterMana)	Constructor. Initializes the MonsterDetail instance with the provided monster attributes and sets up the UI components.

Methods

Name	Description

+ void setMonsterName(String monsterName)	Sets the monsterName field with the provided value and formats the text.
+ void setMonsterDamage(String monsterDamage)	Sets the monsterDamage field with the provided value and formats the text.
+ void setMonsterDef(String monsterDef)	Sets the monsterDef field with the provided value and formats the text.
+ void setMonsterHp(String monsterHp)	Sets the monsterHp field with the provided value and formats the text.
+ void setMonsterMana(String monsterMana)	Sets the monsterMana field with the provided value and formats the text.

3.1.6 Class MonsterPane extends GridPane

Field

Name	**Description**
# MonsterPane instance	A protected static field representing the singleton instance of the MonsterPane class.

Constructor

Name	**Description**

+ MonsterPane()	Constructor. Initializes the MonsterPane instance.

Methods

Name	**Description**
+ void init()	Updates the monster images displayed in the MonsterPane based on the current state of the player's monsters.
+ void update()	Updates the monster images displayed in the MonsterPane based on the current state of the player's monsters.
+ void handleHover(Base_Monster monster)	Handles the event when the mouse hovers over a monster image in the MonsterPane. Updates the monster detail displayed in the ActionPane.
+ void handleOnClick(Base_Monster monster)	Handles the event when a monster image in the MonsterPane is clicked. Sets the clicked monster as the player's active monster.
# MonsterPane getInstance()	Returns the singleton instance of the MonsterPane class.

3.2 Package mapPane

3.2.1 Class CongratulationPane extends Pane

Methods

Name	**Description**
+ CongratulationPane ()	Constructor. Initializes the CongratulationPane with a background image and a congratulatory text message centered on the pane.

3.2.2 Class MapPane extends StackPane

Field

Name	**Description**
- MapPane instance	Singleton instance of MapPane.
# Thread gameLoop	Thread that runs the game loop.
- volatile boolean running	Volatile boolean to control the state of the game loop.

+ KeyboardInputs keyHandler	Static instance of KeyboardInputs for handling keyboard events.
- Boolean Battle	Boolean indicating if a battle is currently occurring.
- Canvas canvas	Canvas used for drawing the game.
- GraphicsContext gc	GraphicsContext associated with the canvas.
- Player player	Instance of the Player class representing the player in the game.
- static GameMap gameMap	Static instance of GameMap representing the current game map.
- MediaPlayer mediaPlayer	MediaPlayer used for playing background music.

Constructor

Name	**Description**
+ MapPane()	Constructor. Initializes the MapPane, sets up the game loop, starts the background music, and assigns the singleton instance.

Methods

Name	Description
- void update()	Updates the player's position and game state.
- void draw()	Clears the canvas and draws the player and game map.
+ void handleCollideWithRocket()	Handles the event when the player collides with a rocket, stops the game loop, and transitions to a new scene.
+ void handleCollideWithBoss()	Handles the event when the player collides with a boss, stops the game loop, and transitions to a battle scene with fade-out and fade-in effects.
- void init()	Initializes the game state, sets the background image, keyboard events, player, canvas, and generates the game map.
+ static void generateGameMap()	Generates the game map based on the selected map name from MapSelectPane.
+ void createGameLoop()	Creates and starts the game loop.
+ void playMusic()	Plays background music.
+ void stopMusic()	Stops the background music with a fade-out effect.
Getters & Setters	Getter & Setters for all fields

3.2.3 Class MapSelectPane extends GridPane

Field

Name	Description
+ String mapName	Static variable holding the selected map name.

Constructor

Name	Description
+ MapSelectPane()	Constructor. Initializes the MapSelectPane, sets up the grid layout, and initializes the planets.

Methods

Name	Description
- void initializePlanet()	Initializes the planets by creating and configuring ImageView for each planet and adding event handlers to show the planet name.

- ImageView createPlanetView(String imagePath, int width, int height, VPos vPos)	Creates and configures an ImageView for a planet with the specified image path, width, height, and vertical position.
- ImageView createPlanetView(String imagePath, int width, int height, VPos vPos, HPos hPos)	Overloaded method to create and configure an ImageView for a planet with the specified image path, width, height, vertical position, and horizontal position.
- void addPlanetClickHandler(ImageView imageView, String planetName)	Adds a click event handler to the ImageView of a planet to handle the event when a planet is clicked.

3.2.4 Class MapTransitionPane extends Pane

Field

Name	**Description**
# ArrayList<Text> dots	ArrayList to store the Text nodes representing dots in the loading animation.
# int dotSize	Size of each dot in the loading animation.
# Text loading	Text node representing the loading message.
# MediaPlayer mediaPlayer	MediaPlayer for playing transition music.

Constructor

Name	Description
+ MapTransitionPane()	Constructor. Initializes the MapTransitionPane, sets up the background image, loading text, dot animation, and transition to the MapPane.

Methods

Name	Description
- void initializeDot()	Initializes the dots for the loading animation.
- void updateDots()	Updates the position of the dots in the loading animation.
- void setDotAnimation()	Sets the animation for the loading dots.
- void playMusic()	Plays the transition music using a MediaPlayer.

3.2.5 Class RocketPane extends GridPane

Field

Name	Description
- int BTN_WIDTH	Width of the buttons.
- int BTN_HEIGHT	Height of the buttons.
- Button confirmBtn	Button to confirm the rocket launch.
- Button cancelBtn	Button to cancel the rocket launch.
- Text Title	Text node for the title of the rocket confirmation.
- Image BgImage	Background image for the RocketPane.

Methods

Name	Description
- void init()	Initializes the RocketPane by setting up its components, including the title, buttons, and background image.
- void createTitle()	Creates the title text node for the rocket confirmation.
- void initializeButtons()	Initializes the confirm and cancel buttons, including setting their appearance, size, and event handlers.

- void confirmLaunch()	Handles the action when the user confirms the rocket launch, changing the scene to the MapSelectPane and stopping the music.
- void cancelLaunch()	Handles the action when the user cancels the rocket launch, resetting the player position, resuming the game loop, and removing the rocket pane from the map pane.

3.2.6 Class StartPane extends BorderPane

Field

Name	**Description**
- int BTN_WIDTH	Width of the buttons.
- int BTN_HEIGHT	Height of the buttons.
- MediaPlayer mediaPlayer	Media player for the background music.

Methods

Name	**Description**

- void initializeTitle()	Initializes the title text for the StartPane, setting its font, color, stroke, alignment, and position.
- void initilizeStartBtn()	Initializes the "Start" button for the StartPane, setting its style, size, event handler, and position.
- void initilizeExitBtn()	Initializes the "Exit" button for the StartPane, setting its style, size, event handler, and position.
- void playMusic()	Plays the background music for the StartPane.

4. Package inputs

4.1 Class KeyboardInputs implementsEventHandler<KeyEvent>

Fields

Name	**Description**
+boolean up, down, left, right	boolean variables to check if certain keys for

	moving are current being pressed

Methods

Name	**Description**
+void handle(KeyEvent event)	This method is called when a key event occurs, it checks the type of the event and update the boolean variable accordingly

5. Package Items

5.1 Package Base

5.1.1 Abstract Class Base_Item

Fields

Name	**Description**
- Type type	mana or health
- boolean isUsed	check if it used or not
- int power	how does it increase or decrease you health or mana
# Image image	picture when isUsed is false
# Image usedImage	picture when isUsed is true

Constructor

Name	Description
+ Base_Item(int power)	set power

Methods

Name	Description
+ String toString()	return the string of how it works on the monster
+ getters and setters	getters and setters of all fields

5.1.2 Abstract Class Base_Poison extends Base_Item implements Splashable

Constructor

Name	Description
+ Base_Poison(Type type, int power)	set power and type

Methods

Name	Description
+ void use(Base_Monster monster)	damage enemy monsters

5.1.3 Abstract Class Base_Potion extends Base_Item implements Consumable

Constructor

Name	Description
+ Base_Potion(Type type, int power)	set power and type

Methods

Name	Description
+ void use(Base_Monster monster)	heal ally monsters

5.1.4 Interface Consumable

Methods

Name	Description
void use(Base_Monster monster)	method for potion

5.1.5 Interface Splashable

Methods

Name	Description

void use(Base_Monster monster)	method for poison

5.1.6 Enum Type: HEALTH, MANA

5.2 Package Poisons

5.2.1 Class Mega_Health_Poison extends Base_Poison

Constructor

Name	**Description**
+ Mega_Health_Poison()	set power and type

5.3 Package Potions

5.3.1 Class Health_Potion extends Base_Potion

Constructor

Name	**Description**
+ Health_Potion()	set power and type

5.3.2 Class Mana_Potion extends Base_Potion

Constructor

Name	Description
+ Mana_Potion()	set power and type

5.3.3 Class Mega_Health_Potion extends Base_Potion

Constructor

Name	Description
+ Mega_Health_Potion()	set power and type

5.3.4 Class Mega_Mana_Potion extends Base_Potion

Constructor

Name	Description
+ Mega_Mana_Potion()	set power and type

6. Package Main

6.1 Class Main extends Application

Field

Name	**Description**
- Stage primaryStage	Primary stage for the application.
- Main instance	Singleton instance of the Main class.

Methods

Name	**Description**
+ void start(Stage primaryStage)	Entry point for JavaFX application. Sets up the primary stage with a custom font, initial scene, and event listeners to maintain a 16:9 aspect ratio.
+ void changeScene(Pane newPane, Boolean transition)	Changes the scene to a new pane with an optional transition effect.
+ void changeSceneStatic(Pane newPane, Boolean transition)	Static method to change the scene using the singleton instance.
+ void fadeAudio(MediaPlayer mediaPlayer,	Fades out the audio of the given

int durationInSeconds)	MediaPlayer over a specified duration.
+ void main(String[] args)	Main method to launch the application.
+ Main getInstance()	+ Main getInstance()

7. Package map

7.1 Abstract Class GameMap

Field

Name	**Description**
# Boundary boundary	Represents the boundary of the game map.
# Sprite boss	Represents the boss character in the game map.
# Rocket rocket	Represents the rocket in the game map.
# double initialX	Initial X position of the player on the map.
# double initialY	Initial Y position of the player on the map.
# Boolean earthIsCleared	Indicates if Earth is cleared.
# Boolean blackHoleIsCleared	Indicates if the Black Hole is cleared.

# Boolean planet1IsCleared	Indicates if Planet 1 is cleared.
# Boolean planet2IsCleared	Indicates if Planet 2 is cleared.
# Boolean planet3IsCleared	Indicates if Planet 3 is cleared.

Methods

Name	**Description**
# abstract boolean checkBoundary(double x, double y)	Checks if the given coordinates are within the boundary # Boundary getBoundary()of the map.
# Boundary getBoundary()	Returns the boundary of the map.
# abstract void drawBoundary(GraphicsContext gc)	Draws the boundary of the map on the provided GraphicsContext.
# void draw(GraphicsContext gc)	Draws the boss and rocket on the provided GraphicsContext.
# void setBoss(Sprite boss)	Sets the boss for the map if the map is not cleared; otherwise, sets the boss to null.
# Sprite getBoss()	Returns the boss of the map.
# void resetBoss()	Resets the boss's attributes to their

	maximum values.
# Rocket getRocket()	Returns the rocket of the map.
# void setPlayerPosition(double posX, double posY)	Sets the player's position to the specified coordinates.
# void resetPlayerPosition()	Resets the player's position to the initial coordinates.
# void setInitialPosition(double x, double y)	Sets the initial position of the player.
# abstract void setIsCleared(Boolean b)	Sets the cleared status of the map.
# abstract Boolean isCleared()	Returns the cleared status of the map.
# Boolean gameCleared()	Checks if all maps (Earth, Black Hole, Planet 1, Planet 2, and Planet 3) are cleared.

7.2 Class MapBlackHole extends GameMap

Constructor

Name	**Description**
+ MapBlackHole()	Constructor. Initializes the boundary, player position, boss, and rocket for the map.

Methods

Name	**Description**

+ boolean checkBoundary(double x, double y)	Checks if the given coordinates are within the boundary of the map.
+ void drawBoundary(GraphicsContext gc)	Draws the boundary of the map on the provided GraphicsContext.
+ Boolean isCleared()	Returns whether the Black Hole map is cleared.
+ void setIsCleared(Boolean isCleared)	Sets the cleared status of the Black Hole map.

7.3 Class MapEarth extends GameMap

Constructor

Name	**Description**
+ MapEarth()	Constructor. Initializes the boundary, player position, boss, and rocket for the map.

Methods

Name	**Description**
+ boolean checkBoundary(double x, double y)	Checks if the given coordinates are within the boundary of the map.

+ void drawBoundary(GraphicsContext gc)	Draws the boundary of the map on the provided GraphicsContext.
+ Boolean isCleared()	Returns whether the Earth map is cleared.
+ void setIsCleared(Boolean isCleared)	Sets the cleared status of the Earth map.

7.4 Class MapPlanet1 extends GameMap

Field

Name	**Description**
- Boundary boundary2	Additional boundary for the map (ParabolicBoundary).
- Boundary boundary3	Additional boundary for the map (LinearBoundary).

Constructor

Name	**Description**
+ MapPlanet1()	Constructor. Initializes boundaries, player position, boss, and rocket for the map.

Methods

Name	Description
+ boolean checkBoundary(double x, double y)	Checks if the given coordinates are within the map's boundaries.
+ void drawBoundary(GraphicsContext gc)	Draws all boundaries of the map on the provided GraphicsContext.
+ Boolean isCleared()	Returns whether the Planet 1 map is cleared.
+ void setIsCleared(Boolean isCleared)	Sets the cleared status of the Planet 1 map.

7.5 Class MapPlanet2 extends GameMap

Field

Name	Description
- Image tree	Tree’s Image in MapPlanet2.
- Image tree_Rock	Tree with rock’s in MapPlanet2.
- Boundary boundary2	Additional boundary for the map (ParabolicBoundary).
- Boolean isCleared	The clear status of MapPlanet2.

Constructor

Name	Description

+ MapPlanet2()	Constructor. Initializes boundaries, player position, boss, and rocket for the map.

Methods

Name	**Description**
+ boolean checkBoundary(double x, double y)	Checks if the given coordinates are within the map's boundaries.
+ void drawBoundary(GraphicsContext gc)	Draws all boundaries of the map on the provided GraphicsContext.
+ void draw(GraphicsContext gc)	Draw a map components.
+ Boolean isCleared()	Returns whether the Planet 2 map is cleared.
+ void setIsCleared(Boolean isCleared)	Set the clear status of Planet 2 map.

7.6 Class MapPlanet3 extends GameMap

Field

Name	**Description**
- Image component	Image representing a component of the Planet 3 map.

Constructors

Name	**Description**
+ MapPlanet3()	Constructor. Initializes boundary, player position, boss, and rocket for the map.

Methods

Name	**DEscription**
+ boolean checkBoundary(double x, double y)	Checks if the given coordinates are within the map's boundary.
+ void drawBoundary(GraphicsContext gc)	Draws the boundary of the map on the provided GraphicsContext.
+ void draw(GraphicsContext gc)	Draws the map components, boss, and rocket on the provided GraphicsContext.
+ Boolean isCleared()	Returns whether the Planet 3 map is cleared.
+ void setIsCleared(Boolean isCleared)	Sets the cleared status of the Planet 3 map.